Hebbian Iterative Method for Unsupervised Clustering with Automatic Detection of the Number of Clusters with Discrete Recurrent Networks

Enrique Mérida-Casermeiro and Domingo López-Rodríguez

Department of Applied Mathematics
University of Málaga, Málaga, Spain
{merida, dlopez}@ctima.uma.es

Abstract. In this paper, two important issues concerning pattern recognition by neural networks are studied: a new model of hebbian learning, as well as the effect of the network capacity when retrieving patterns and performing clustering tasks. Particularly, an explanation of the energy function when the capacity is exceeded: the limitation in pattern storage implies that similar patterns are going to be identified by the network, therefore forming different clusters.

This ability can be translated as an unsupervised learning of pattern clusters, with one major advantage over most clustering algorithms: the number of data classes is automatically learned, as confirmed by the experiments. Two methods to reinforce learning are proposed to improve the quality of the clustering, by enhancing the learning of patterns relationships.

As a related issue, a study on the net capacity, depending on the number of neurons and possible outputs, is presented, and some interesting conclusions are commented.

1 Introduction

In 1949, Hebb [2], introduced a physiological learning method based on the reinforcement of the interconnection strength between neurons. It was explained in the following terms:

> When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.

This kind of learning method has been widely applied to recurrent networks in order to store and retrieve patterns in terms of their similarity. Models that used this learning rule were the bipolar model (BH) presented by J. J. Hopfield in 1982 [4], representing a powerful neural model for content addressable memory, or its analog version [5], among others. These networks, although successful in

R. Marín et al. (Eds.): CAEPIA 2005, LNAI 4177, pp. 241–250, 2006.

solving many combinatorial optimization problems, present two main problems when used as content-addressable memory: their low capacity and the apparition of spurious patterns.

The capacity parameter α is usually defined as the quotient between the maximum number of patterns to load into the network and the number of used neurons that obtains an acceptable error probability (usually $p_{\text{error}} = 0.05$ or 0.01). It has been shown that this constant is approximately $\alpha = 0.15$ for BH.

This value means that, in order to load K patterns, more than $\frac{K}{\alpha}$ neurons will be needed to achieve an error probability equal to p_{error}. Or equivalently, if the net is formed by N neurons, the maximum number of patterns that can be loaded in the net (with that error constraint) is $K < \alpha N$.

Recently, in [8], a multivalued recurrent network is used to avoid the undesirable apparition of spurious patterns by using the so-called *augmented patterns* (the method presented in that work is also valid for BH). Patterns are correctly retrieved if sufficiently separated.

The main idea of this work holds that when patterns are very close each other, or if the net capacity is exceeded, then local minima corresponding to similar patterns tend to be combined, forming one unique local minimum. So, although considered as a limitation of the net as associative memory, this fact can explain the way in which the human brain form concepts: several patterns, all of them similar to a common typical representative, are associated (as in the vector quantization), and form a group in which particular features are not distinguishable.

Obviously, enough samples are needed to generalize and not to distinguish their particular features. If there exist few samples from some class, they will still be retrieved by the net individually, that is, as an associative memory.

2 MREM Model with Semi-parallel Dynamics

Let $\mathcal{H}$ be a recurrent neural network formed by N neurons, where the state of each neuron i is defined by its output V_i, $i \in \mathcal{I} = \{1, 2, \ldots, N\}$ taking values in any finite set $\mathcal{M} = \{m_1, m_2, \ldots, m_L\}$. This set does not need to be numerical.

The state of the network, at time t, is given by a N-dimensional vector, $\boldsymbol{V}(t) = (V_1(t), V_2(t), \ldots, V_N(t)) \in \mathcal{M}^N$. Associated to every state vector, an energy function, characterizing the behavior of the net, is defined:

$$E(\boldsymbol{V}) = -\frac{1}{2} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} w_{ij} f(V_i, V_j) + \sum_{i \in \mathcal{I}} \theta_i(V_i) \tag{1}$$

where $w_{i,j}$ is the weight of the connection from the j-th neuron to the i-th neuron, $f : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ can be considered as a measure of similarity between the outputs of two neurons, usually verifying the similarity conditions mentioned in [8,10]:

1. For all $x \in \mathcal{M}$, $f(x, x) = c \in \mathbb{R}$.
2. f is a symmetric function: for every $x, y \in \mathcal{M}$, $f(x, y) = f(y, x)$.
3. If $x \neq y$, then $f(x, y) \leq c$.

and $\theta_i : \mathcal{M} \to \mathbb{R}$ are the threshold functions. Since thresholds will not be used for content addressable memory, henceforth we will consider θ_i be the zero function for all $i \in \mathcal{I}$.

The introduction of this similarity function provides, to the network, of a wide range of possibilities to represent different problems [6,7,8,9,10,11]. So, it leads to a better representation than other multivalued models, as SOAR and MAREN [1,12], since in those models most of the information enclosed in the multivalued representation is lost by the use of the signum function that only produces values in $\{-1, 0, 1\}$.

In every instant, the net evolves to reach a state of lower energy than the current one.

In this work, we have considered discrete time and semi-parallel dynamics, where only one neuron is updated at time t. The next state of the net will be the one that achieves the greatest descent of the energy function by changing only one neuron output.

Let us consider a total order in $\mathcal{M}$. The potential increment when a-th neuron changes its output from V_a to $l \in \mathcal{M}$ at time t, is $U_{a,l}(t) = -\Delta E$:

$$U_{a,l}(t) = \frac{1}{2} \sum_{i \in \mathcal{I}} [w_{a,i} f(l, V_i(t)) + w_{i,a} f(V_i(t), l) - (w_{a,i} f(V_a(t), V_i(t)) + \\ + w_{i,a} f(V_i(t), V_a(t)))] - \frac{1}{2} w_{aa} [f(l, l) - f(V_a(t), V_a(t))] \quad (2)$$

If f verifies the similarity conditions, then the *reduced potential increment* is obtained:

$$U^*_{a,l}(t) = -\frac{1}{2} \sum_{i \in \mathcal{I}} [(w_{a,i} + w_{i,a}) \cdot (f(V_i(t), l) - f(V_i(t), V_a(t)))] \quad (3)$$

We use the following updating rule for the neuron outputs:

$$V_a(t+1) = \begin{cases} l, & \text{if } U_{a,l}(t) \geq U_{b,k}(t) \forall k \in \mathcal{M} \text{ and } \forall b \in \mathcal{I} \\ V_a(t), & \text{otherwise} \end{cases} \quad (4)$$

This means that each neuron computes in parallel the value of a L-dimensional vector of potentials, related to the energy decrement produced if the neuron state is changed. The only neuron changing its current state is the one producing the maximum decrease of energy.

It has been proved that the MREM model with this dynamics always converges to a minimal state [7]. This result is particularly important when dealing with combinatorial optimization problems, where the application of MREM has been very fruitful [6,7,8,9,10,11].

3 MREM as Auto-Associative Memory

Now, let $X = \{\boldsymbol{X}_k : k \in \mathcal{K}\}$ be a set of patterns to be loaded into the neural network. Then, in order to store a pattern, $\boldsymbol{X} = (x_i)_{i=1,2,\ldots,N}$, components of the W matrix must be modified in order to make $\boldsymbol{X}$ the state of the network with minimal energy.

As pointed out in [8,10], since energy function is defined as in Eq. (1), we calculate $\frac{\partial E}{\partial w_{ij}} = -\frac{1}{2} f(V_i, V_j)$ and we modify the components of matrix W in order to reduce the energy of state $\boldsymbol{V} = \boldsymbol{X}$ by the rule $\Delta w_{i,j} = -\alpha \frac{\partial E}{\partial w_{i,j}} = \frac{\alpha}{2} f(x_i, x_j)$ for some $\alpha > 0$.

Particularly, for $\alpha = 2$ (every $\alpha > 0$ produces functionally equivalent networks, see [8,10]), it results:

$$\Delta w_{i,j} = f(x_i, x_j) \tag{5}$$

and considering that, at first, $W = 0$, that is, all the states of the network have the same energy and adding over all the patterns, the next expression is obtained:

$$w_{i,j} = \sum_{k \in \mathcal{K}} f(x_{ki}, x_{kj}) \tag{6}$$

Equation (6) is a generalization of *Hebb's postulate of learning*, because the weight w_{ij} between neurons is increased in correspondence with their similarity.

It must be pointed out that, when bipolar neurons and the product function are used, $f(x, y) = xy$, the well-known learning rule of patterns in the Hopfield's network is obtained.

In order to recover a loaded pattern, the network is initialized with the known part of that pattern. The network dynamics will converge to a stable state (due to the decreasing of the energy function), that is, a minimum of the energy function, and it will be the answer of the network. Usually this stable state is next to the initial one.

But, as explained in [8,10] in terms of similarity between vectors associated to different states, when the bipolar Hopfield network loads the pattern $\boldsymbol{X}$, by Eq. (5) with $f(V_i, V_j) = V_i V_j$, the energy of other states is also reduced and the opposite state is also loaded. States with local minimum energy and no associated with input patterns are called *spurious states* and loading spurious states is usually considered an undesirable effect.

4 Spurious Patterns

In [8,10], a technique to avoid the apparition of spurious states when loading pattern $\boldsymbol{X} = (x_i)$ in BH is presented.

Definition 1. *Given a state $\boldsymbol{V}$ of the net, and a similarity function f, its associated matrix ($G_{\boldsymbol{V}}$) is defined as a $N \times N$ matrix whose elements are $G_{i,j} = f(V_i, V_j)$.*

In addition, it can be defined its associated vector ($\boldsymbol{G_V}$) as the vector with N^2 components obtained by expanding matrix $G_{\boldsymbol{V}} = (G_{i,j})$ into vector form, $\boldsymbol{G_V} = (G_k)$, verifying $G_{j+N(i-1)} = G_{i,j}$.

Definition 2. *Suppose that $\boldsymbol{X} = (x_1, x_2, \ldots, x_N)$ is a pattern to be loaded in the net, and $\mathcal{M} = \{m_1, m_2, \ldots, m_L\}$. The* **augmented pattern** *associated to $\boldsymbol{X}$ is the vector $\hat{\boldsymbol{X}} = (x_1, x_2, \ldots, x_N, m_1, m_2, \ldots, m_L) \in \mathcal{M}^{N+L}$ whose components are $\hat{x_i} = x_i$ if $i \leq N$ and $\hat{x_i} = m_{i-N}$ if $i > N$.*

It must be pointed out that, in order to load augmented patterns into the net, only N neurons are necessary, since last L components are clamped to a fixed value. It is only necessary to consider the weights $w_{i,j}$ associated to these components.

Theorem 1. *The function $\Psi : \hat{\boldsymbol{X}} \to \boldsymbol{G}_{\hat{\boldsymbol{X}}}$, that associates every augmented pattern to its associated vector, is injective.*

Proof. Please refer to [8].

So, instead of loading a pattern $\boldsymbol{X}$ with $x_i \in \mathcal{M} = \{m_1, m_2, \ldots, m_L\}$, its associated augmented pattern $\hat{\boldsymbol{X}}$ can be loaded into the net. The state $\hat{\boldsymbol{X}}$ will be the only one maximizing the energy decrease, since the term $W_{\hat{\boldsymbol{X}}} = (f(\hat{x}_i, \hat{x}_j))_{i,j}$ is added to the previously computed weight matrix W, as indicated by Eq. (6).

5 Some Remarks on the Capacity of the Net

In [8], Mérida et al. find an expression for the capacity parameter α for MREM model in terms of the number of neurons N and the number of possible states for each neuron, L, for the case in which N is big enough to apply the Limit Central Theorem ($N \geq 30$):

$$\alpha(N, L) \approx \frac{1}{N} + \frac{\left(\frac{A^2}{{z_\alpha}^2} - B\right)}{NC} \tag{7}$$

where $A = N + 3 + \frac{(N-1)(4-L)}{L}$, $B = \frac{8(N-1)(L-2)}{L^2}$, $C = \frac{8N}{L}$ and z_α is obtained by imposing the condition that the maximum allowed error probability in retrieving patterns is p_{error}. For $p_{\text{error}} = 0.01$, we get $z_\alpha \approx 2.326$.

Some facts can be extracted from the above expression.

For a fixed number of neurons, capacity is not bounded above: Suppose N fixed. Equation (7) can be rewritten in the following form:

$$\alpha(N, L) \approx \frac{1}{N^2 z_\alpha^2} \left[2L + 4(N-1) + z_\alpha^2 + \frac{2(N-1)(N-1+z_\alpha^2)}{L}\right]$$

If we make L tend to ∞, we get $\lim_{L\to\infty} \alpha(N, L) = \infty$, since the coefficient of L in this expression is positive.

What actually happens is that $\alpha(N, \cdot)$, as a function of L, has a minimum at the point $L_0(N) = \sqrt{(N-1)(N-1+z_\alpha^2)} \approx N + 1$ for $z_\alpha = 2.326$. It is a decreasing function for $L < L_0(N)$ and increasing for $L \geq L_0(N)$.

One consequence of this result is that, for appropriate choice of N and L, the capacity of the net can be $\alpha(N, L) > 1$.

This fact can be interpreted as a adequate representation of the multivalued information, because, to represent the same patterns as MREM with N and L fixed, BH needs NL binary neurons and therefore the maximum number of stored patterns may be greater than N. So it is not a strange thing that the capacity

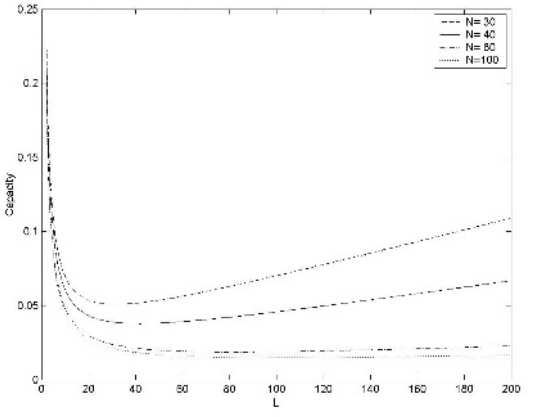

Fig. 1. Representation of the capacity α of the network versus the number of possible states of the neurons, L, for different values of N, the number of neurons

can reach values greater than 1, if the patterns are multivalued, MREM needs much less neurons to represent the pattern than BH.

For a fixed number of possible outputs, capacity is bounded below by a positive constant: Suppose L is fixed. Equation (7) can be rewritten as follows:

$$\alpha(N,L) \approx \frac{1}{z_\alpha^2 L}\left[2 + \frac{4(L-1)+2z_\alpha^2}{N} + \frac{2(L-1)^2+(L-2)z_\alpha^2}{N^2}\right]$$

It can be easily seen that this expression represents a function whose value decreases as N grows. So, a net with more neurons than other, and the same possible states, will present less capacity than the second one. Thus, a minimum positive capacity can be computed for each possible value of L, verifying $\alpha_{\min}(L) = \lim_{N\to\infty} \alpha(N,L) = \frac{2}{z_\alpha^2 L} > 0$.

$\alpha_{\min}(L)$ coincides with the asymptotic capacity for the net with L possible neuron outputs. For example, if $L = 2$ (as in BH), an asymptotic capacity of $\alpha_{\min}(2) = 0.1847$ is obtained, exactly the capacity for BH provided in other works [3].

All these facts can be observed in Fig. 1, representing the value of the parameter $\alpha(N,L)$ for $N \in \{30, 40, 80, 100\}$ and $L \in \{2, 3, \ldots, 200\}$.

6 When Capacity is Exceeded

This work tries to explain what may happen psychologically in the human brain. When a reduced number of patterns has to be memorized, the brain is able to remember all of them when necessary. Similarly, when the capacity of the net is not exceeded, the net is able to retrieve exactly the same patterns that were loaded into it. But when the brain receives a great amount of data to be recognized or classified, it distinguishes between some groups of data (in an unsupervised

way). This kind of behavior is also simulated by neural networks, as we will show next, proving the power (and adequation) of the model herein presented.

Then, learning rules as Hebb's (or the more general given by Eq. (5)), where connection between neurons is reinforced by the similarity of their expected outputs, may produce classifiers that discover some knowledge from the input patterns, like the actual number of groups in which the data is divided. Then, an unsupervised clustering of the input pattern space is automatically performed.

If a pattern, say $\boldsymbol{X}$, is to be loaded in the net, by applying Eq. (5), a local minimum of the energy function E is created at $\boldsymbol{V} = \boldsymbol{X}$. If another pattern $\boldsymbol{X'}$ is apart from $\boldsymbol{X}$, its load will create another local minimum. But, if $\boldsymbol{X}$ and $\boldsymbol{X'}$ are close each other, these two local minima created by the learning rule will be merged, forming one local minima instead.

Then, if a group of patterns is loaded into the net (overflowing its capacity), and all of them are close each other, only one local minimum will be formed, and at the moment of retrieving these data, the unique pattern to be retrieved will be associated to the state of minimum energy. So, patterns can be classified by the stable state of the net which they converge to.

This technique has also the advantage of time. Its execution time only depends on the number of patterns, and it is totally independent of the number of classes in which the data is divided into.

7 Learning Reinforcement

Equation (5) for the learning rule, generalization of Hebb's one, shows that the only thing taking part in updating weight matrix is the pattern to be loaded into the net at that time. So, it represents a very 'local' information, and does not take account of the possible relationships that pattern could have with the already stored ones. So, it is convenient to introduce an additional mechanism in the learning phase, such that the information concerning to relationships between patterns is incorporated in the update of the weight matrix. In what follows, we will consider that the similarity function is $f(x, y) = 2\delta_{x,y} - 1$, that is, its value is 1 if $x = y$ and -1 otherwise.

Learning Reinforcement Method: Suppose that we have the (augmented) pattern $\boldsymbol{X}_1$ stored in the net. So, we have the weight matrix $W = (w_{i,j})$. If pattern $\boldsymbol{X}_2$ is to be loaded into the network, by applying Eq. (5), components of matrix ΔW are obtained.

If $w_{i,j}$ and $\Delta W_{i,j}$ have positive signum (both values equal 1), it means that $X_{1i} = X_{1j}$ and $X_{2i} = X_{2j}$, indicating the relationship between X_{1i}, X_{1j}, X_{2i} and X_{2j}. If both are negative valued, something similar happens, but with inequalities instead of equalities.

So, the fact of $w_{i,j}$ and $\Delta W_{i,j}$ having the same signum is a clue of a relationship that is repeated between components i and j of patterns $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$. In order to reinforce the learning of this relationship, we propose a novel technique, presenting also another kind of desirable behavior: The model proposed before, given by Eq. (6), is totally independent of the order in which patterns

are presented to the net. This fact does not actually happen in the human brain, since every new information is analyzed and compared to data and concepts previously learned and stored.

So, to simulate this kind of learning, a method named **LR** is presented:

Let us multiply by a constant, $\beta > 1$, the components of matrices W and ΔW where the equality of signum is verified, i.e., the components verifying $w_{i,j} \cdot \Delta W_{i,j} > 0$. Hence the weight matrix learned by the network is, after loading pattern $\boldsymbol{X}_2$:

$$w'_{i,j} = \begin{cases} w_{i,j} + \Delta W_{i,j} & \text{if } w_{i,j} \cdot \Delta W_{i,j} < 0 \\ \beta[w_{i,j} + \Delta W_{i,j}] & \text{if } w_{i,j} \cdot \Delta W_{i,j} > 0 \end{cases} \tag{8}$$

Similarly, if there are some patterns $\{\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_R\}$ already stored in the network, in terms of matrix W, and pattern $\boldsymbol{X}_{R+1}$ is to be loaded, matrix ΔW (corresponding to $\boldsymbol{X}_{R+1}$) is computed and then the new learning rule given by Eq. (8) is applied.

It must be noted that this method satisfy Hebb's postulate of learning quoted in Sec. 1.

This learning reinforcement technique, **LR**, has the advantage that it is also possible to learn patterns one by one or by blocks, by analyzing at a time a whole set of patterns, and comparing the resulting ΔW to the already stored in the net. Then, for instance, if $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_R\}$ has already been loaded into the net in terms of matrix W, we can load a whole set $\{\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_M\}$ by computing $\Delta W = (\sum_{k=1}^{M} f(y_{ki}, y_{kj}))_{i,j}$ and then applying Eq. (8).

Iterative Learning Reinforcement Method: LR can be improved in many ways. In this section, an iterative approach, **ILR**, to enhance the solution given by **LR** is presented.

Suppose that, by using Eq. (8) of **LR**, matrix W_X related to pattern set $X = \{\boldsymbol{X}_k : k \in \mathcal{K}\}$ has been learned and denote by $\varphi_{W_X}(\boldsymbol{X}_k)$ the stable state reached by the network (with weight matrix W_X) when beginning from the initial state given by $\boldsymbol{V} = \boldsymbol{X}_k$.

Then, the cardinal of $\{\varphi_{W_X}(\boldsymbol{X}_k) : k \in \mathcal{K}\}$ is (no multiplicities) the number of classes that **LR** finds, n_{a}.

$\varphi_{W_X}(X) := \{\varphi_{W_X}(\boldsymbol{X}_k) : k \in \mathcal{K}\}$ can be considered (with all multiplicities included) as a new pattern set, formed by a noiseless version of patterns in X. So, if applying a second time **LR** to $\varphi_{W_X}(X)$, by using Eq. (8) to build a new matrix $W_{\varphi_{W_X}(X)}$, better results are expected than in the first iteration, since the algorithm is working with a more refined pattern set.

This whole process can be repeated iteratively until a given stop criterion is satisfied. For example, when two consecutive classifications assign each pattern to the same cluster.

8 Simulations

In order to show the ability of **LR** and **ILR** to perform a clustering task as mentioned in Sec. 6, several simulations have been made whose purpose is the clustering of discrete data.

Table 1. Average clustering results on 10 runs of these algorithms, where n_a indicates the number of obtained clusters, p_{cc} is the correct classification percentage (that is, the percentage of simulations in which $n_a = n$) and Err. is the average error percentage

	$n=3$						$n=4$						$n=5$					
	LR			**ILR**			**LR**			**ILR**			**LR**			**ILR**		
K	n_a	p_{cc}	Err.	n_a	p_{cc}	Err.	n_a	p_{cc}	Err.	n_a	p_{cc}	Err.	n_a	p_{cc}	Err.	n_a	p_{cc}	Err.
75	3.1	90	0.53	3.1	90	0.53	4.7	60	3.46	4.1	90	0.26	6.0	40	7.33	5.2	80	2.00
150	3.3	70	0.33	3.1	90	0.13	4.8	70	1.06	4.4	80	0.73	7.5	20	4.00	6.2	30	2.80
300	3.7	70	0.60	3.3	80	0.43	5.5	30	3.06	4.6	60	4.80	7.7	0	3.16	5.8	50	0.50
450	3.5	70	0.37	3.1	90	0.17	4.9	60	0.55	4.4	80	0.31	7.2	10	4.93	6.0	40	4.44
600	3.2	80	0.25	3.2	80	0.25	5.4	50	0.61	4.6	50	0.31	8.4	30	3.03	7.0	50	2.88
750	3.3	80	0.06	3.0	100	0.00	5.4	30	3.49	4.6	60	3.12	10.6	10	9.88	8.1	10	5.56
900	3.2	90	3.23	3.0	100	0.00	5.5	20	0.56	4.8	40	0.42	8.3	10	3.30	6.3	40	2.76
Av.	3.3	78	0.76	3.1	90	0.21	5.2	46	1.82	4.5	66	1.42	7.9	17	5.09	6.3	43	2.99

Several datasets have been created, each of them formed by K 50-dimensional patterns randomly generated around n centroids, whose components were integers in the interval $[1, 10]$. That is, the n centroids were first generated and input patterns were formed from them by introducing some random noise modifying one component of the centroid with probability 0.018. So, the Hamming distance between input patterns and the corresponding centroids is a binomial distribution $B(50, 0.018)$. Patterns are equally distributed among the n clusters. It must be noted that patterns may have Hamming distance even 5 or 6 from their respective centroid, and new clusters can be formed by this kind of patterns.

So, a network with $N = 50$ neurons taking value in the set $\mathcal{M} = \{1, \ldots, 10\}$ has been considered. The parameter of learning reinforcement has been chosen $\beta = 1.5$. It has been observed that similar results are obtained for a wide range of values of β.

The results obtained in our experiments are shown in Table 1. It can be observed not only the low classification error (from 0% to 9.88% on average), but in addition these new techniques get the exact, or very approximate, number of groups in which the pattern set is actually divided in almost every simulation. In fact, whenever the number n_a of discovered clusters equals n, an error percentage of 0% is obtained, retrieving in those cases the initial centroids. It can also be verified that **ILR** clearly outperforms **LR** in most cases, getting a more accurate classification and improving the estimation of the number of clusters, as seen in the last row of the table.

9 Conclusions

In this work, we have explained that the limitation in capacity of storing patterns in a recurrent network has not to be considered as determinant, but it can be used for the unsupervised classification of discrete patterns.

The neural model MREM, based on multivalued neurons, has been developed, as a generalization of the discrete Hopfield model and as an auto-associative memory, improving some of the undesirable aspects of the original Hopfield model: some methods and results for the net not to store spurious patterns in the learning phase have been shown, reducing so the number of local minima of the energy function not associated to an input pattern.

By applying a slight modification to Hebb's learning rule, a mechanism to reinforce the learning of the relationships between different patterns has been introduced to the first part of the process, incorporating knowledge corresponding to several patterns simultaneously. This mechanism can be repeated iteratively to enhance the relationship learning procedure.

In addition, simulations confirm the idea expressed in this work, getting optimal results of classification in many cases.

This work presents a new open research line, from the fact that some new modifications of the learning rule, reinforcing other aspects of the relationships among patterns, may be developed.

References

1. M. H. Erdem and Y. Ozturk, *A New family of Multivalued Networks*, Neural Networks **9,6**, 979-989, 1996.
2. D. O. Hebb, *The Organization of Behavior*, New York: Wiley, 1949.
3. J. Hertz, A. Krogh and R. G. Palmer, *Introduction to the theory of neural computation*, Lecture Notes Volume I. Addison Wesley, 1991.
4. J.J. Hopfield, *Neural networks and physical systems with emergent collective computational abilities*, Proc. of National Academy of Sciences USA, **79**, 2254-2558, 1982.
5. J. J. Hopfield, *Neurons with graded response have collective computational properties like those of two-state neurons*, Proceedings of the National Academy of Sciences USA, **81**, 3088-3092, 1984.
6. D. López-Rodríguez and E. Mérida-Casermeiro, *Matrix Bandwidth Minimization: A Neural Approach*, ICCMSE, **1**, 324-327, 2004.
7. E. Mérida Casermeiro, *Red Neuronal recurrente multivaluada para el reconocimiento de patrones y la optimización combinatoria*, Ph. D. dissertation (in spanish). Univ. Málaga, España, 2000.
8. E. Mérida Casermeiro and J. Muñoz-Pérez, *MREM: An associative autonomous recurrent network*, Journal of Intelligent and Fuzzy Systems 12(3-4), 163-173, 2002.
9. E. Mérida Casermeiro, J. Muñoz Pérez and R. Benítez Rochel, *A recurrent multivalued neural network for the N-queens problem*, Lecture Notes in Computer Science **2084**, 522-529, 2001.
10. E. Mérida Casermeiro, J. Muñoz-Pérez and M. A. García-Bernal, *An Associative Multivalued Recurrent Network*, IBERAMIA 2002, 509-518, 2002.
11. E. Mérida-Casermeiro and D. López-Rodríguez, *Multivalued Neural Network for Graph MaxCut Problem*, ICCMSE, **1**, 375-378, 2004.
12. Y. Ozturk and H. Abut, *System of associative relationships (SOAR)*, Proceedings of ASILOMAR, 1997.